

Comparing GPT-4.1 and Claude Sonnet 4.6 for Automated Retrieval of SEC Financial Filings from Corporate Investor Relations Websites

Immersive Software Engineering LM173

Final Year Report

07/04/2026

Student Name	Daniel Moody
Student ID	23370157
Supervisor	Salaheddin Alakkari

Contents

1 Introduction **3**

1.1 Background and Motivation . . . 3
1.2 Residency Context . . . 3
1.3 Research Questions . . . 4
1.4 Literature Review . . . 5
1.4.1 LLM Agentic Capabilities and Web Navigation . . . 5
1.4.2 Web Crawling: Static Fetching, Bot Detection, and JavaScript Rendering . . . 6
1.4.3 Corporate Financial Filings: EDGAR and Global IR Disclosure . . . 7
1.4.4 LLM Benchmarking Methodology . . . 8

2 Methodology **10**

2.1 Overview . . . 10
2.2 Dataset and Ground Truth Construction . . . 10
2.2.1 Company Selection . . . 10
2.2.2 Ground Truth Verification . . . 10
2.2.3 Alternative URL Handling . . . 11
2.3 Retrieval System Design . . . 11
2.3.1 HTTP Fetching Layer . . . 11
2.3.2 Two-Phase LLM Navigation . . . 12
2.3.3 IR Subdomain Fallback . . . 13
2.4 Prompt Design . . . 13
2.5 Evaluation Metrics . . . 14
2.5.1 Primary Metrics . . . 14
2.5.2 Secondary Metrics . . . 14
2.6 Ethical Considerations . . . 15
2.7 Test Environment . . . 15
2.8 Benchmark Procedure . . . 16

3 Quantitative Results **19**

3.1 Overall Exact Match Rates . . . 19
3.2 Run-to-Run Variance . . . 20
3.3 Outcome Breakdown . . . 20
3.4 IR Subdomain Fallback Activation . . . 22
3.5 Autonomous LLM Navigation to SEC EDGAR . . . 22
3.6 Secondary Metric Summary . . . 23
3.7 Cost and Latency . . . 23
3.8 Head-to-Head Divergence . . . 25

4 Discussion **27**
4.1 Form 10-K Retrieval: Claude Sonnet 4.6 is Clearly Stronger 27
4.2 Form 10-Q Retrieval: GPT-4.1 is Marginally Stronger, Claude More Stable 27
4.3 IR Subdomain Fallback: High Precision, Uneven Reliance 28
4.4 Autonomous LLM Navigation to SEC EDGAR 28
4.5 JavaScript Rendering as the Dominant Failure Constraint 29
4.6 Cost and Latency Trade-offs 30
4.7 Limitations 30
4.8 Threats to Validity 31

5 Conclusion **32**
5.1 Summary of Findings 32
5.2 Recommendations 32
5.3 Future Work 33

Declaration of Academic Integrity **35**

AI Use Declaration **35**

A Prompt Templates **38**

B Benchmark Runner **39**

C JavaScript Failure Evidence **40**

1 Introduction

1.1 Background and Motivation

The systematic analysis of publicly listed companies requires access to a large volume of structured financial documentation. For companies listed on US exchanges, the primary regulatory filings are the Form 10-K, an annual report filed with the Securities and Exchange Commission (SEC), and the Form 10-Q, a quarterly update filed three times per year. These documents contain the financial statements, management commentary, and footnote disclosures that form the quantitative and qualitative foundation of investment research. Analysts use them to construct financial models, evaluate management performance, and assess the competitive positioning of the businesses they cover.

Acquiring these documents at scale presents a practical challenge. Although the SEC maintains a centralised public filing repository, EDGAR (Electronic Data Gathering, Analysis, and Retrieval), many companies also publish their filings directly on their own investor relations (IR) websites, often in a more accessible format and with supplementary materials. For investment research workflows that require the most recent filing in direct document form, the IR site is frequently the preferred source. The difficulty is that no two IR sites are structured alike. Some companies host filings on dedicated investor subdomains; others embed filing links within their corporate site navigation; others serve documents through content delivery network infrastructure with opaque URLs. Navigating this heterogeneous landscape manually does not scale across a large universe of companies.

Large language models (LLMs) offer a potential solution. Their ability to parse natural language page content, follow contextual navigation cues, and identify relevant links makes them candidates for automating the traversal of IR sites without requiring site-specific scraping rules. However, the practical effectiveness of different LLM APIs for this task has not been systematically evaluated. The choice of model has direct implications for the accuracy, cost, and latency of any production retrieval system.

1.2 Residency Context

This project was conducted during a Residency 4 placement at a quantitative investment research firm. The firm operates an AI-assisted research pipeline in which analyst questions about public companies are answered using information drawn from a variety of document sources, including earnings call transcripts, news archives, and corporate filings. At the time of the placement, the pipeline lacked an automated component for acquiring corporate annual and quarterly reports from IR sites. Research analysts were reliant on manual document retrieval, which represented a bottleneck in the broader information

ingestion process.

The project brief, as discussed with the supervising engineer, was to build and evaluate a system capable of navigating from a company's root website URL to the correct filing document on its IR site, with AI assistance to handle the variability in site structure. The supervising engineer explicitly framed this as a benchmarking exercise, noting that multiple AI model options should be evaluated to identify the most suitable choice for production use, and that a dataset of known-correct filing URLs should be assembled to enable objective comparison.

In response to this brief, a dataset of 99 publicly listed US companies was constructed, with ground truth filing URLs for both the Form 10-K and Form 10-Q manually verified by the author. A shared retrieval system was developed to provide a controlled execution environment for both models under evaluation, and a series of benchmark runs was conducted across all four conditions defined by the combination of model and filing type.

The findings of this study are directly applicable to the firm's production system design decision. The model that demonstrates the strongest combination of retrieval accuracy and cost efficiency on this benchmark is the recommended candidate for integration into the live document acquisition pipeline.

1.3 Research Questions

This study addresses the following research questions:

1. Does GPT-4.1 or Claude Sonnet 4.6 achieve a higher exact match rate when retrieving Form 10-K filing documents from corporate IR websites, and what factors explain any observed difference?
2. Does GPT-4.1 or Claude Sonnet 4.6 achieve a higher exact match rate when retrieving Form 10-Q filing documents from corporate IR websites, and is the ordering consistent with the 10-K condition?
3. What site-level and model-level factors most strongly predict retrieval failure across both models and filing types?
4. What are the cost and latency trade-offs between the two models, and how do these interact with their respective accuracy profiles to inform a production deployment recommendation?

1.4 Literature Review

1.4.1 LLM Agentic Capabilities and Web Navigation

Large language models have evolved considerably beyond static text generation. The emergence of multi-step reasoning and tool-use capabilities has made it possible to deploy LLMs as autonomous agents that interact with external environments over sequences of decisions. The ReAct framework (Yao et al. 2023) formalised one influential approach to this paradigm: at each step the model generates a natural language reasoning trace before issuing a structured action, interleaving thought and behaviour in a loop that continues until a terminal condition is reached. This reason-then-act structure is directly instantiated in the retrieval system described in this study. At each navigation step the LLM receives the Markdown-rendered content of the current page, reasons about which link is most likely to lead toward the target document, and issues one of three structured actions: `FOUND`, `NEXT`, or `NONE`. The correspondence to the ReAct loop is precise, with the page content serving as the observation and the structured action serving as the act.

The application of LLMs to web-based information retrieval has prior precedent. Nakano et al. (2021) demonstrated that a language model equipped with a web browsing interface could answer factual questions by searching, following links, and extracting relevant passages, establishing the fundamental viability of LLM-guided web navigation as a research area. A limitation of the WebGPT approach, however, is its reliance on a full browser interface for each interaction, which introduces per-page latency that is prohibitive for production systems required to process large numbers of sites efficiently; this constraint directly motivates the static-fetch architecture adopted in the present study. More recently, Deng et al. (2023) introduced Mind2Web, a benchmark of real-world web tasks requiring agents to navigate heterogeneous websites by identifying and interacting with the correct elements at each step. Their findings highlight the difficulty of generalising across diverse site structures, a challenge directly relevant to the IR site heterogeneity encountered in this study. However, both WebGPT and Mind2Web primarily evaluate broad web-task competence rather than strict direct-document retrieval. They demonstrate that an agent can navigate plausibly across heterogeneous websites, but they do not directly evaluate the operational distinction that matters here: reaching the correct filing entry is not sufficient if the agent stops on an index or listing page rather than the filing document itself.

A critical enabling mechanism for reliable multi-step automation is structured output generation. Rather than producing free-form text, both models in this study were required to conform to a `NavigationDecision` schema specifying three fields: action, URL, and a confidence message. This constraint ensures that the system can parse and act on the model's output deterministically at each step, without requiring post-hoc extraction from

natural language. In this study, both models were constrained to return outputs matching this schema via the structured output capabilities of their respective APIs.

A known limitation of LLMs in agentic settings is their tendency to hallucinate plausible-sounding but incorrect outputs. In the navigation context this manifests as returning index pages or filing search results rather than direct document URLs, despite explicit prompt instructions to return a direct file link. This instruction-following failure mode is observed in the benchmark results and is discussed in Section 4.1.

1.4.2 Web Crawling: Static Fetching, Bot Detection, and JavaScript Rendering

Automated retrieval of web content presents three distinct engineering challenges: avoiding bot detection systems, handling JavaScript-rendered content, and managing the performance cost of doing so at scale. The design of the retrieval system in this study reflects a deliberate set of trade-offs across all three.

TLS client implementations can be fingerprinted at the handshake level. Standard HTTP client libraries therefore often present TLS signatures that differ from real browser processes, creating a network-level signal by which automated traffic can be distinguished from mainstream browser traffic (Frolov & Wustrow 2019). To address this, the retrieval system uses the `curl-cffi` library configured to impersonate a Chrome 131 browser at the TLS layer, producing a handshake that is much closer to genuine browser traffic. This approach was selected after evaluating available bypass strategies as the most effective method for reducing fingerprint-based blocking without the overhead of a full browser runtime.

The standard engineering alternative for both bot detection bypass and JavaScript rendering is a full browser automation framework such as Playwright or Puppeteer. These tools drive a real browser process, producing authentic TLS fingerprints and executing client-side JavaScript to populate page content. However, for a production system required to crawl large numbers of companies efficiently, the per-page overhead of launching and managing a browser process is prohibitive. Each page fetch via Playwright requires initialising a browser context, waiting for the page to fully render, and tearing down the context, adding several seconds of latency per page independent of network conditions. The decision to use static fetching in this study was therefore a deliberate production performance constraint, not an oversight; Playwright was evaluated during the system design phase and rejected on these grounds. This also exposes a limitation in much prior browser-agent work: results achieved with full browser execution are not automatically transferable to production pipelines that must process large company universes under strict latency and cost constraints. A browser-centric evaluation can therefore overstate

the operational feasibility of web-agent retrieval at scale.

The consequence of this choice is that pages relying on client-side JavaScript to populate their content are inaccessible to the retrieval system. The challenge of crawling JavaScript-heavy web applications is well-documented in the web crawling literature. Mirtaheri et al. (2013) survey the evolution of web crawling techniques and show how crawler design has repeatedly had to adapt to shifts in web architecture. Mesbah et al. (2012) formalise the problem for Ajax-based applications, showing that a static fetch of a single-page application returns only the application shell rather than the populated content, since the data is loaded asynchronously after the initial HTML response. IR portals built on this architecture therefore return an effectively empty document to the retrieval system, preventing the LLM from identifying any filing links regardless of its navigational capability. The implications of this constraint for the benchmark results are examined in Section 4.5.

1.4.3 Corporate Financial Filings: EDGAR and Global IR Disclosure

The United States Securities and Exchange Commission operates EDGAR (Electronic Data Gathering, Analysis, and Retrieval), a centralised mandatory filing repository for all US-listed public companies (U.S. Securities and Exchange Commission 2024). Form 10-K and Form 10-Q submissions are publicly searchable through EDGAR using company identifiers such as the Central Index Key (CIK), and each filing resolves to a stable filing page and associated document set. This makes EDGAR an authoritative source for ground truth verification: for any given company and filing type, the correct document URL can be independently confirmed through EDGAR even when the primary retrieval target is the company's own IR site.

The academic use of EDGAR and 10-K text as a data source for financial research is extensive. Loughran & McDonald (2011) showed that generic text-analysis dictionaries can misclassify financially meaningful language in 10-K filings, underscoring both the richness and the domain specificity of filing text. More recent work has applied language models directly to financial text corpora: Yang et al. (2020) pre-trained a BERT-based model on large-scale financial communications, and Wu et al. (2023) developed BloombergGPT, a large language model trained on a wide range of financial data for finance-related tasks. This body of work underscores the significance of filings and related financial text as primary information sources in financial NLP and the relevance of automated retrieval at scale.

A limitation of EDGAR-based retrieval is the CIK resolution problem. When a model autonomously constructs or follows an EDGAR URL, it must correctly identify the CIK associated with the target company. An incorrect CIK results in the retrieval of filings for a different entity, a failure mode that is silent at the URL-return stage but detectable only

by comparing the filing content to the ground truth. This failure mode was observed in the benchmark results for Claude Sonnet 4.6 on the Form 10-Q condition, where several autonomous EDGAR navigations returned filings for incorrect companies.

Outside the United States, centralised filing repositories either do not exist or vary significantly by jurisdiction. Many markets require companies to publish regulatory filings only on their own investor relations websites, with no standardised URL structure or metadata convention. The system evaluated in this study is designed for this global deployment context, where direct IR site navigation is the only reliable retrieval path. US companies were used for this benchmark because EDGAR provides a robust, independently verifiable ground truth, making objective evaluation straightforward. The structural heterogeneity of US IR sites, which ranges from dedicated investor subdomains to CDN-hosted documents, is representative of the variation encountered in global IR site navigation, making the benchmark results applicable to the broader deployment use case.

1.4.4 LLM Benchmarking Methodology

Rigorous evaluation of large language models requires controlled experimental conditions, well-defined metrics, and sufficient repetition to account for output variance. Several established benchmarks have shaped the conventions used in this study.

Hendrycks et al. (2021) introduced MMLU (Massive Multitask Language Understanding), a benchmark covering 57 subject areas evaluated via multiple-choice questions with fixed ground truth answers. The use of exact match against a curated ground truth set as the primary evaluation metric, applied uniformly across all conditions with no model-specific adjustment, is a principle adopted directly in this study. Chen et al. (2021) applied the same principle to code generation, evaluating whether generated programs pass a fixed set of unit tests; the binary pass/fail metric is analogous to the exact URL match metric used here. It should be noted, however, that both MMLU and HumanEval evaluate single-turn, closed-form tasks; the multi-hop web navigation task in this study involves sequential decisions over an unpredictable number of steps, making it inherently more variable and less directly comparable to static question-answering benchmarks. Both benchmarks also reward correctness in tightly bounded answer spaces, whereas this study evaluates open-ended URL retrieval in which many plausible but operationally wrong answers are possible.

For retrieval tasks specifically, Bajaj et al. (2016) introduced MS MARCO, a large-scale machine reading comprehension and passage-ranking dataset derived from real search queries. The exact match paradigm for retrieval evaluation, where a returned item is scored against a manually verified ground truth without partial credit, is standard across this class of benchmarks and is the evaluation approach adopted in this study.

MS MARCO still evaluates retrieval over a pre-indexed corpus with provided passages, whereas the task in this study requires the model to navigate across multiple live pages with no prior index, making it a substantially harder retrieval problem that is not directly captured by existing benchmarks.

A known methodological concern in LLM evaluation is output variance. Because LLM generation is stochastic, repeated evaluation of the same input does not always produce the same output, and a single-run result may not be representative of a model's true capability on a given task. Liang et al. (2022) addressed this concern in the HELM benchmark by evaluating models across multiple scenarios and metrics, arguing that broad, multi-dimensional evaluation produces more reliable comparisons than narrow single-metric studies. This study follows the same general principle by conducting three independent runs per condition and reporting means and standard deviations, with the standard deviation on the GPT-4.1 10-K condition (4.20 percentage points) providing a concrete illustration of why single-run results should be treated with caution. HELM is still only a partial fit for the present task, however, because it does not focus on repeated multi-hop document retrieval from live and changing websites. Existing benchmark conventions therefore provide useful evaluation principles, but not a task-specific template for exact-match filing retrieval on heterogeneous IR infrastructure.

Beyond accuracy, production deployments of LLM systems must evaluate cost and latency alongside correctness. A model that achieves higher accuracy at substantially greater API cost may not be preferable in all deployment contexts, particularly where retrieval is performed at scale across large company universes. This study therefore reports mean per-company API cost and wall-clock retrieval time for each condition, alongside the primary exact match rate, to support a complete production deployment recommendation.

2 Methodology

2.1 Overview

This study employs a controlled quantitative benchmarking design to compare two large language model (LLM) APIs on the task of automated retrieval of SEC financial filings from corporate investor relations (IR) websites. The two systems under evaluation are OpenAI GPT-4.1 (model identifier: `gpt-4.1`) and Anthropic Claude Sonnet 4.6 (model identifier: `claude-sonnet-4-6`). A purpose-built retrieval system was developed to provide a shared, controlled execution environment for both models, ensuring that any observed differences in outcome can be attributed to the models themselves rather than to differences in tooling, prompting strategy, or evaluation criteria. Both models were accessed exclusively via their respective REST APIs. No fine-tuning, system-prompt customisation, or model-level configuration was applied beyond the task-level prompts described in Section 2.4.

2.2 Dataset and Ground Truth Construction

2.2.1 Company Selection

The benchmark dataset comprises 99 publicly listed US companies drawn from a range of industries and market sectors. US companies were selected for this benchmark for a specific methodological reason: because all US-listed companies are required to file with the SEC's EDGAR repository, every ground truth URL can be independently verified against a centralised authoritative source, making objective evaluation straightforward. The system under evaluation is designed for global deployment, where no equivalent centralised repository exists and companies publish filings exclusively on their own IR sites; however, the structural heterogeneity of US IR sites is representative of the global landscape, making US companies a suitable proxy population for benchmarking purposes.

Companies were selected to ensure diversity in IR site architecture, encompassing organisations that host their filings on dedicated investor subdomain portals (e.g. `investors.company.com`), those with IR sections embedded within their primary corporate website, and those that serve filing documents directly via content delivery network (CDN) infrastructure such as Amazon CloudFront. This diversity was intentional: a dataset composed exclusively of straightforward IR sites would not provide a meaningful assessment of model robustness across real-world conditions.

2.2.2 Ground Truth Verification

For each company, two ground truth URLs were identified and manually verified prior to any benchmark run:

- The direct document URL of the most recent **Form 10-K** (the SEC annual report filing).
- The direct document URL of the most recent **Form 10-Q** (the most recent SEC quarterly report filing).

Ground truth URLs were required to point to the actual filing document (a PDF or equivalent direct download), not to a filing index page or an intermediate navigation page. This distinction is deliberate and is discussed further in Section 2.5. All ground truth URLs were stored alongside each company's root domain URL in a structured CSV file (`benchmark_companies.csv`). The CSV contains six columns: `company`, `root_url`, `url_10k`, `alt_url_10k`, `url_10q`, and `alt_url_10q`.

2.2.3 Alternative URL Handling

In certain cases, an identical filing document is accessible under more than one valid URL. For example, a company may serve the same PDF from both a corporate domain path and an IR subdomain path, or a filing may be accessible via both an `http` and an `https` address. To prevent such cases from being recorded as false negatives, an `alt_url_10k` and `alt_url_10q` column were included in the ground truth CSV. A retrieval result was deemed correct if the returned URL matched either the primary ground truth URL or the alternative URL, after normalisation.

2.3 Retrieval System Design

A custom retrieval system was developed to serve as the shared evaluation harness for both models. The system operates as an LLM-guided web crawler and is composed of three functional components: an HTTP fetching layer, a two-phase LLM navigation engine, and an IR subdomain fallback mechanism.

2.3.1 HTTP Fetching Layer

Web pages are retrieved using the `curl-cffi` library (version 0.14.0) configured to impersonate a Chrome 131 browser at the TLS handshake level. This impersonation strategy allows the crawler to bypass fingerprint-based bot detection mechanisms that would otherwise block standard HTTP client requests, without requiring a full browser runtime. Upon retrieval, HTML responses are converted to Markdown format using the `html2text` library (version 2025.4.15) before being passed to the LLM. This conversion preserves hyperlink structure while substantially reducing the token volume required to represent a page, thereby reducing API cost and latency.

Page content is truncated to a maximum of 100,000 characters (approximately 25,000 tokens) prior to being passed to the LLM. This ceiling was chosen to remain within the

context window limits of both models while avoiding the cost and latency penalty of processing unusually large pages in full.

This fetching strategy is static by design: no JavaScript execution is performed and no browser automation is used. Pages are retrieved as the raw HTTP response body only. Browser automation tools such as Playwright were evaluated during the system design phase but rejected on performance grounds; the per-page overhead of launching and managing a browser process is prohibitive for a production system required to crawl large numbers of companies efficiently. The curl-cffi approach with Chrome TLS impersonation was selected as the best available trade-off between bot detection bypass capability and retrieval speed. The implications of this constraint for sites that rely on client-side JavaScript rendering are discussed in Section 4.5.

2.3.2 Two-Phase LLM Navigation

The core navigation logic follows a two-phase design. In the **navigation phase**, the LLM is provided with the Markdown-rendered content of the company's root URL and instructed to identify and follow links toward the company's investor relations section. The LLM issues one of three structured actions at each step: `FOUND` (the target page has been reached, with the URL provided), `NEXT` (a promising unvisited link to follow is identified, with the URL provided), or `NONE` (no further candidate links remain). This loop continues iteratively until the LLM returns `FOUND` or `NONE`, or until a maximum traversal depth of 15 hops is reached.

Upon successful completion of the navigation phase, the **document search phase** begins from the identified IR page. The same iterative loop is applied, but the objective is now to identify and return the direct download URL of the target filing document, rather than a navigation or listing page. The LLM is explicitly instructed to return a direct file URL (e.g. a PDF link) and not a filing index or search results page.

At each step in both phases, previously visited URLs are tracked and injected into the prompt to prevent the LLM from revisiting pages and entering an infinite loop. Loop detection is also performed at the system level: if the LLM selects a URL that has already been visited, navigation terminates immediately with a `loop` termination reason rather than re-fetching the page.

The structured output schema used at each step is a `NavigationDecision` object containing three fields: `action` (one of `FOUND`, `NEXT`, or `NONE`), `url` (nullable), and `confidence_message` (a free-text explanation of the decision). Both models were required to conform to this schema via their respective structured output APIs.

2.3.3 IR Subdomain Fallback

A subset of corporate websites do not expose any navigable link to their investor relations section from the homepage. In such cases the navigation phase terminates with a `NONE` or `fetch_failed` outcome, returning no document URL. To handle this class of failure, the system implements an IR subdomain fallback mechanism.

When the primary navigation phase fails to return a document URL, the system concurrently probes five common IR subdomain patterns by substituting the company's registered domain into each template:

- `investor.{domain}`
- `investors.{domain}`
- `ir.{domain}`
- `corporate.{domain}`
- `stock.{domain}`

Each candidate URL is probed via a lightweight HTTP GET request. Any subdomain that returns a successful response is then used as the starting point for the document search phase directly, bypassing the navigation phase entirely. The fallback returns the first successful result found across all resolving subdomains. If no subdomain resolves or none yields a document URL, the original failure result is returned unchanged.

2.4 Prompt Design

Each filing type was associated with a fixed `DocumentQuery` object specifying three parameters: the document type label, a recency filter, and a set of disambiguation hints. The disambiguation hints were designed to resolve common sources of confusion observed during system development.

For the **Form 10-K** condition, the hints instructed the model to target the SEC regulatory annual filing specifically, and to disregard links labelled only as "Annual Report" unless they explicitly referenced the Form 10-K. This distinction is necessary because many companies publish both a glossy shareholder annual report and a separate Form 10-K, and the former is typically more prominently linked from IR pages.

For the **Form 10-Q** condition, the hints instructed the model to target the most recent quarterly regulatory filing and to disregard earnings press releases, investor presentations, and earnings call transcripts, all of which are commonly listed alongside filings on IR pages and share similar visual presentation.

The same prompt templates and disambiguation hints were used for both GPT-4.1 and

Claude Sonnet 4.6 across all runs, ensuring that any observed performance difference is attributable to model behaviour rather than to differential prompting. The full prompt template and `DocumentQuery` definitions are provided in Appendix A.

2.5 Evaluation Metrics

Each retrieval attempt was assigned one of three outcome statuses prior to metric computation:

- **Success**: the system returned a non-empty document URL.
- **Failed**: the system terminated all navigation attempts without returning any URL.
- **Error**: an unhandled exception occurred after all retry attempts were exhausted.

Five quantitative metrics were recorded per company per run and used to evaluate and compare the two models.

2.5.1 Primary Metrics

Success rate is the proportion of the 99 companies for which the system returned any non-empty URL, regardless of correctness. It captures the system's ability to complete the retrieval task but does not distinguish between correct and incorrect results.

Exact match rate is the proportion of the 99 companies for which the returned URL, after normalisation, matched the manually verified ground truth URL (primary or alternative). URL normalisation consisted of converting the scheme to `https` and removing any trailing forward slash prior to string comparison. Exact match rate is the primary evaluation metric throughout this study. A high success rate accompanied by a low exact match rate indicates that the model is returning plausible but incorrect URLs, such as filing index pages rather than direct document URLs. Because the task prompt explicitly instructs the model to return the direct filing document, returning an index page is classified as an instruction-following failure rather than a navigation failure.

2.5.2 Secondary Metrics

LLM call count is the number of API calls made to the language model per company retrieval. Each navigation step in both the navigation phase and the document search phase constitutes one LLM call. This metric reflects the navigational complexity encountered per company and is a direct driver of API cost and latency.

Total token consumption is the sum of input and output tokens consumed per company retrieval. Input tokens comprise the rendered page content and prompt; output tokens

comprise the structured navigation decision. Together with the per-token pricing of each model, this metric determines the per-company API cost.

Wall-clock retrieval time is the total elapsed time in seconds from the start of a company's crawl to the return of a result, including all HTTP fetch operations, LLM API round-trips, and retry delays. This metric reflects end-to-end operational latency.

Table 1 summarises descriptive secondary-metric values across the full three-run dataset.

Table 1: Descriptive secondary metric values per company across all four conditions over three runs ($n = 297$ per condition).

Metric	**GPT 10-K**	**Claude 10-K**	**GPT 10-Q**	**Claude 10-Q**
LLM calls (mean)	4.17	4.37	5.25	5.90
LLM calls (median)	3	3	4	4
LLM calls (max)	55	23	78	24
Total tokens (mean)	19,241	30,774	24,439	40,266
Wall-clock time (mean, s)	12.9	22.3	14.6	29.8
Wall-clock time (median, s)	8.9	14.8	10.7	18.9

2.6 Ethical Considerations

All web requests made during benchmarking were directed at publicly accessible investor relations pages and SEC EDGAR, both of which are intended for unrestricted public access. No authentication was bypassed, no private data was accessed, and request rates were kept low through sequential per-company processing with retry delays, ensuring no material load was placed on any target server.

2.7 Test Environment

All benchmark runs were conducted on a single workstation to ensure consistency of the hardware and network environment across conditions. The system specification is given in Table 2.

Table 2: Test environment specification.

Component	Specification
Operating system	Windows 11 Business, Version 25H2 (OS build 26200.8037)
System architecture	64-bit operating system on ARM-based processor
Processor	Snapdragon X 12-core X1E80100 @ 3.42 GHz
Memory	16 GB RAM (15.6 GB usable)
Graphics	Qualcomm Adreno X1-85 GPU
Python version	3.14.3
`curl-cffi`	0.14.0
`html2text`	2025.4.15
`openai` SDK	2.9.0
`anthropic` SDK	0.86.0

Both models were accessed via their respective cloud APIs; no local model weights were used. API credentials were loaded from environment variables at runtime. No shared session state was maintained between consecutive company crawls within a run. Each company was processed sequentially. If an exception occurred, the company crawl was retried automatically up to three times, with a three-second delay between attempts.

Because the benchmark depends on third-party cloud APIs, additional run-quality controls were applied to reduce distortion from transient service instability. In the Claude client, Anthropic overload responses were handled by waiting 30 seconds and retrying the affected request. Isolated overload responses were therefore absorbed in-line and the benchmark continued normally. When overload responses became unusually frequent, the Claude benchmark was paused and resumed later during a lower-demand period. This was done to avoid treating vendor-side service instability as model retrieval failure, so only completed runs obtained under acceptable service conditions were retained in the final dataset.

2.8 Benchmark Procedure

The study adopts a 2 x 2 factorial design, producing four benchmark conditions defined by the combination of model (GPT-4.1 or Claude Sonnet 4.6) and filing type (Form 10-K or Form 10-Q). Each condition was executed across three independent full runs over all 99 companies, yielding 297 company-runs per condition and 1,188 company-runs in total. Results for each company were written to a structured JSON file immediately upon completion of that company's crawl, ensuring that no data was lost in the event of an interruption to the run.

Each completed company crawl was recorded as a structured JSON object with a stable schema. This schema was designed to support both per-company error analysis and aggregate benchmark reporting. Table 3 summarises the principal fields stored for each company record. A final summary object appended at the end of each file records the aggregate success rate, exact match rate, and total token usage for that condition and run number.

Table 3: Principal fields recorded in each benchmark JSON result object.

Field	**Description**
`company`	Company name used for benchmark identification.
`root_url`	Starting corporate website URL supplied to the crawler.
`document_type`	Filing target for the run (Form 10-K or Form 10-Q).
`status`	Outcome class for the attempt: `success`, `failed`, or `error`.
`doc_url`	Final document URL returned by the system, if any.
`expected_url`	Manually verified primary ground truth URL for that company and filing type.
`url_match`	Boolean exact-match result after URL normalisation and alternative-URL checking.
`used_subdomain_fallback`	Boolean indicating whether the coded IR subdomain fallback was activated.
`doc_termination_reason`	Terminal reason reported by the document search stage, such as `found`, `none_action`, or `loop`.
`model`	Model identifier used for the run.
`metrics`	Nested object containing timing and control-flow statistics such as fetch count, LLM call count, and total elapsed time.
`token_usage`	Nested object containing per-company input, output, and total token consumption.

Three complete runs were conducted for each of the four conditions. All per-condition metrics reported in this study are means across these three runs, with standard deviations computed to characterise inter-run variability. This repeated-run design was used to reduce the risk of drawing conclusions from a single stochastic navigation trace and to distinguish stable model behaviour from run-specific noise on difficult IR sites.

This repeated-run design was especially important for the Claude conditions, where transient API overload could otherwise be misclassified as retrieval failure. The included runs therefore represent completed benchmark passes judged to reflect model behaviour under acceptable service conditions rather than periods of abnormal vendor-side instability.

3 Quantitative Results

3.1 Overall Exact Match Rates

Table 4 presents the per-run and mean exact match results for all four benchmark conditions. Each run consists of $n = 99$ independent company retrievals, giving $n = 297$ company-runs per condition across the three-run study. Exact match rate is the primary evaluation metric, measuring whether the returned URL corresponds exactly to the manually verified ground truth document URL after normalisation.

Table 4: Exact match rates across three runs per condition ($n = 99$ per run).

Model	Filing Type	Run 1	Run 2	Run 3	Mean	SD (pp)
GPT-4.1	Form 10-K	74.75%	68.69%	66.67%	70.04%	4.20
Claude Sonnet 4.6	Form 10-K	82.83%	79.80%	77.78%	80.14%	2.54
GPT-4.1	Form 10-Q	74.75%	75.76%	71.72%	74.08%	2.10
Claude Sonnet 4.6	Form 10-Q	72.73%	71.72%	72.73%	72.39%	0.58

Across the three-run study, Claude Sonnet 4.6 outperforms GPT-4.1 on Form 10-K retrieval by 10.10 percentage points on a mean basis (80.14% vs 70.04%). On Form 10-Q retrieval the ordering reverses, but the gap is much smaller: GPT-4.1 achieves a mean exact match rate of 74.08% compared to 72.39% for Claude Sonnet 4.6, a difference of 1.69 percentage points. Figure 1 presents these results visually.

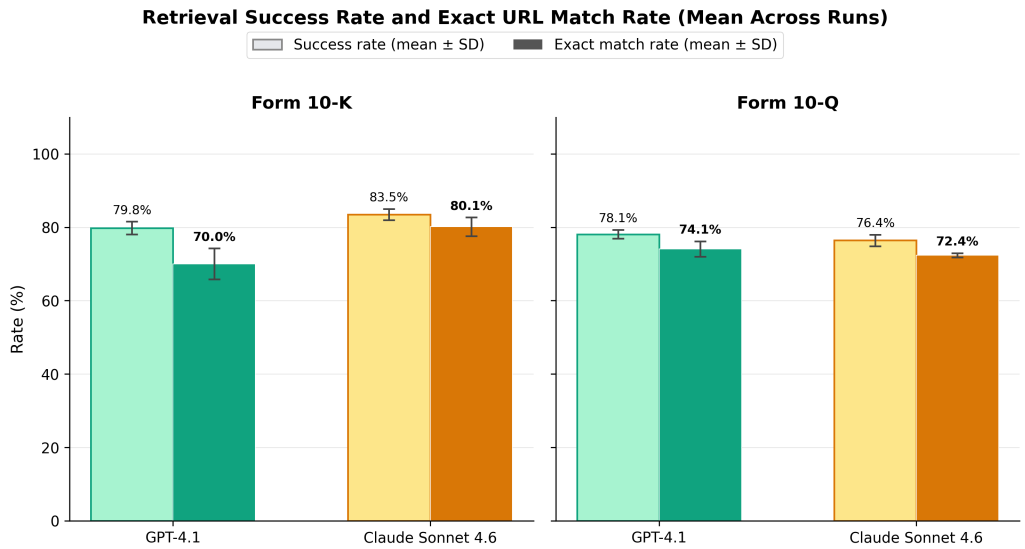


Figure 1: Mean success and exact match rates per condition across three runs ($n = 297$ company-runs per condition). Error bars show standard deviation across runs.

The gap between success rate and exact match rate is largest for GPT-4.1 on the 10-

K condition. Across 297 company-runs, GPT-4.1 returned a non-empty URL in 79.8% of attempts but matched the exact document in only 70.0%, meaning only 87.8% of its successful retrievals were exact matches. Claude Sonnet 4.6 returned a non-empty URL in 83.5% of 10-K attempts and achieved an exact match rate of 80.1%, corresponding to an exact-match-of-success rate of 96.0%. On the 10-Q condition, by contrast, both models were much closer in final-document precision: GPT-4.1 achieved an exact-match-of-success rate of 94.8% and Claude 94.7%. The quarterly difference is therefore driven more by reach and cost than by final-document selection quality.

3.2 Run-to-Run Variance

The standard deviations in Table 4 show that annual retrieval is more variable than quarterly retrieval, particularly for GPT-4.1. GPT-4.1's 10-K exact match rate declines from 74.75% in Run 1 to 66.67% in Run 3, producing the largest standard deviation in the study at 4.20 percentage points. Claude Sonnet 4.6 also drifts downward on 10-K, but less sharply (SD = 2.54). On the quarterly condition, GPT-4.1 remains comparatively stable (SD = 2.10), while Claude Sonnet 4.6 is the most stable condition in the benchmark, varying by only 0.58 percentage points across the three runs.

Figure 2 shows the per-run exact match rate for each condition. The annual gap is stable in direction across all runs, whereas the quarterly gap remains narrow and reverses slightly in Run 3.

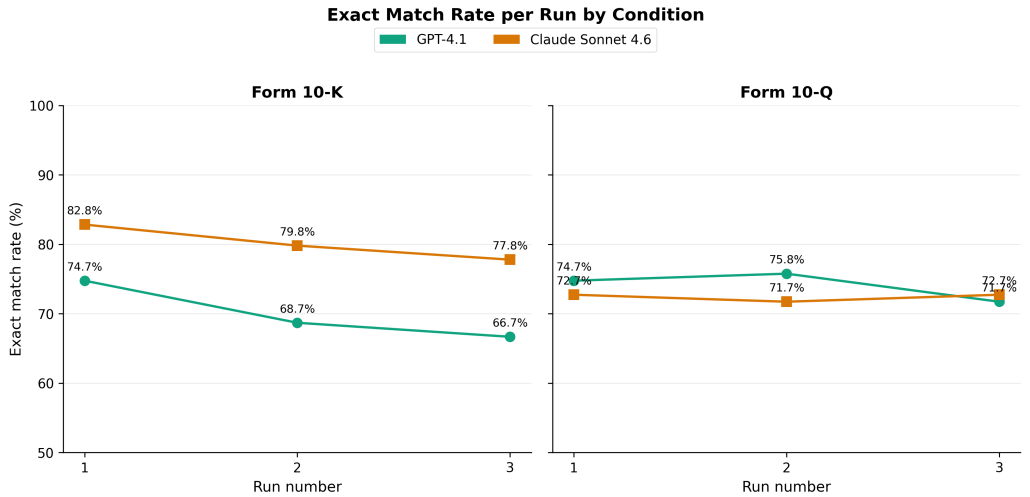


Figure 2: Exact match rate per run for each condition across the three-run study.

3.3 Outcome Breakdown

Figure 3 shows the outcome breakdown across all four conditions. Outcomes are classified as success (a non-empty URL was returned), failed (all navigation attempts terminated

without a URL), or error (an unhandled exception occurred after all retries were exhausted).

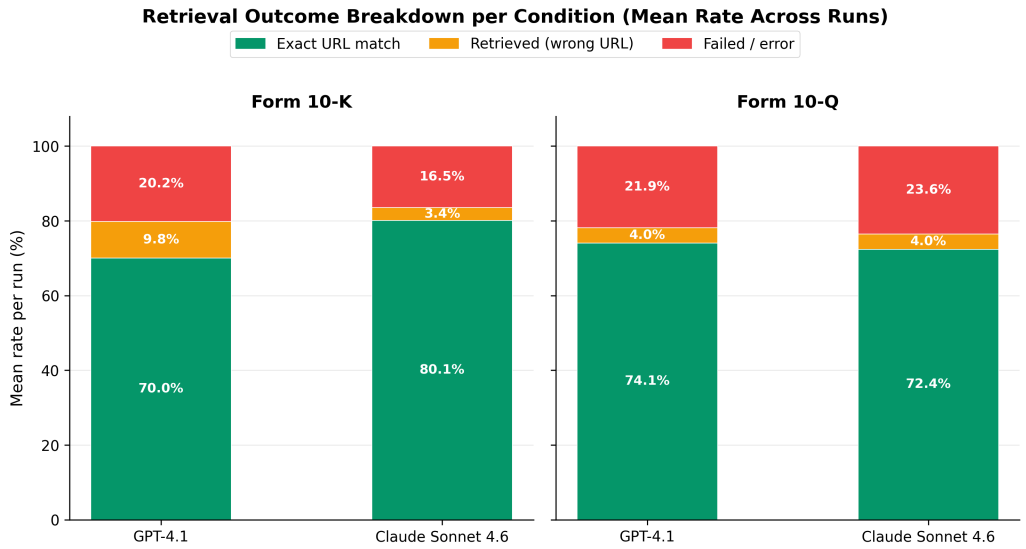


Figure 3: Mean outcome breakdown (success, failed, error) per condition across three runs.

Errors are rare across all four conditions. The dominant differences are therefore not infrastructure crashes but the balance between successful retrieval, failed reach, and successful-but-incorrect outcomes. Figure 4 further breaks down the failure categories, distinguishing between navigation failures, document search failures, and other termination reasons.

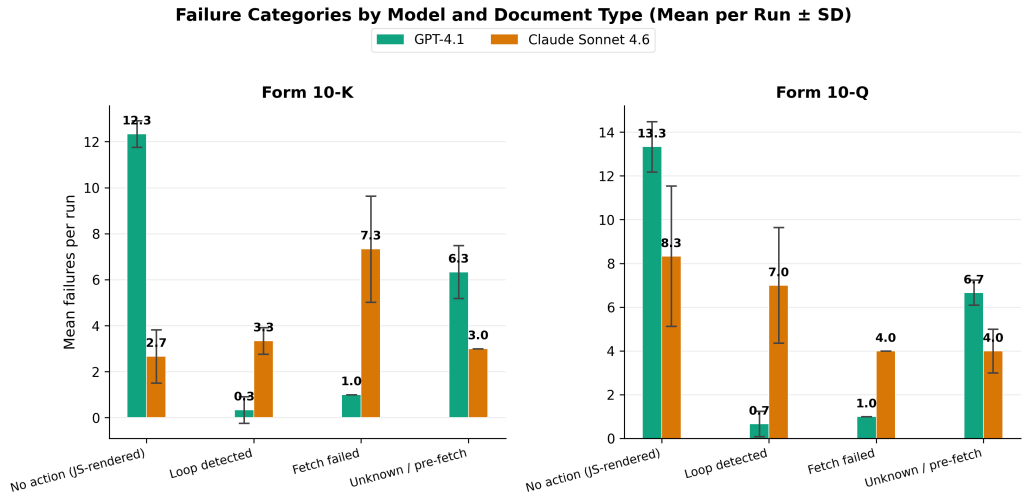


Figure 4: Mean failure category breakdown per condition across three runs.

3.4 IR Subdomain Fallback Activation

Table 5 summarises the activation and effectiveness of the coded IR subdomain fallback mechanism across the three runs. This fallback activates when the primary navigation phase fails to return any document URL, and probes five common IR subdomain patterns for the company's domain as described in Section 2.3.

Table 5: IR subdomain fallback correctness by run. Values are correct retrievals / fallback activations.

Model	Filing Type	Run 1	Run 2	Run 3	Combined
GPT-4.1	Form 10-K	15 / 15 (100.0%)	9 / 10 (90.0%)	10 / 11 (90.9%)	34 / 36 (94.4%)
Claude Sonnet 4.6	Form 10-K	10 / 10 (100.0%)	7 / 9 (77.8%)	9 / 9 (100.0%)	26 / 28 (92.9%)
GPT-4.1	Form 10-Q	14 / 14 (100.0%)	11 / 11 (100.0%)	11 / 11 (100.0%)	36 / 36 (100.0%)
Claude Sonnet 4.6	Form 10-Q	5 / 6 (83.3%)	8 / 9 (88.9%)	10 / 10 (100.0%)	23 / 25 (92.0%)

The subdomain fallback demonstrated high precision across all conditions. GPT-4.1 achieved 94.4% correctness on Form 10-K and 100% on Form 10-Q, while Claude Sonnet 4.6 achieved 92.9% on Form 10-K and 92.0% on Form 10-Q. GPT-4.1 also activated the fallback more often than Claude in both filing types (36 vs 28 activations on 10-K; 36 vs 25 on 10-Q), suggesting that Claude more often reaches the relevant IR area without requiring a coded recovery path.

3.5 Autonomous LLM Navigation to SEC EDGAR

In addition to the coded IR subdomain fallback, analysis of the returned document URLs revealed that both models independently navigated to SEC EDGAR (`sec.gov`) in a subset of cases, without any explicit instruction or coded logic directing them to do so. This behaviour represents an autonomous model decision to seek the filing from the public EDGAR database rather than from the company's own IR site.

Table 6 summarises the frequency and accuracy of this behaviour across all three runs. Claude Sonnet 4.6 exhibited this behaviour substantially more often than GPT-4.1 across both filing types.

Table 6: Autonomous LLM navigation to SEC EDGAR across three runs. Values are navigations / exact matches.

Model	Filing Type	Run 1	Run 2	Run 3	Mean navigations	Mean correct
GPT-4.1	Form 10-K	2 / 0	1 / 0	2 / 0	1.67	0.00
Claude Sonnet 4.6	Form 10-K	10 / 10	12 / 10	9 / 8	10.33	9.33
GPT-4.1	Form 10-Q	2 / 2	2 / 2	2 / 1	2.00	1.67
Claude Sonnet 4.6	Form 10-Q	14 / 13	13 / 11	11 / 11	12.67	11.67

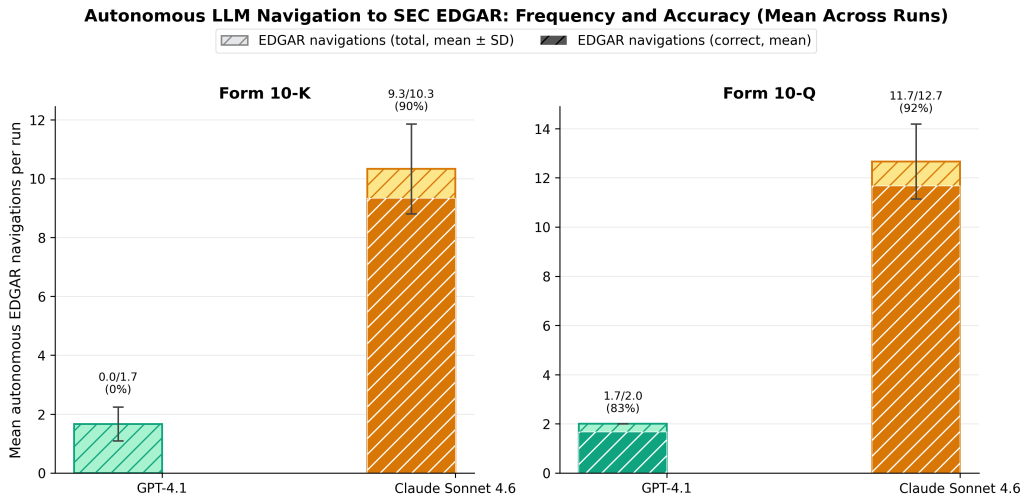


Figure 5: Mean frequency and accuracy of autonomous LLM navigation to SEC EDGAR per condition across three runs. Error bars show standard deviation across runs.

Claude Sonnet 4.6 used EDGAR 31 times on Form 10-K and returned the correct document in 28 of those cases, making EDGAR a meaningful annual recovery path. On Form 10-Q, Claude used EDGAR 38 times and was exact in 35 cases. GPT-4.1 used EDGAR far more sparingly: 5 annual EDGAR retrievals produced 0 exact matches, whereas 5 of its 6 quarterly EDGAR retrievals were exact. The behaviour is therefore both model-specific and filing-type-specific, and is examined further in Section 4.4.

3.6 Secondary Metric Summary

Table 7 summarises the main secondary metrics across the three-run study for each condition. Mean LLM calls, mean input tokens, and mean wall-clock time are calculated over successful retrievals only, whereas mean API cost is reported per benchmark attempt across all 297 observations in each condition.

Table 7: Secondary metric summary across three runs.

Model	Filing Type	Mean LLM Calls	Mean Input Tokens	Mean Cost (USD)	Mean Time (s)
GPT-4.1	Form 10-K	3.59	17,822	$0.04068	11.17
Claude Sonnet 4.6	Form 10-K	3.86	27,390	$0.10134	18.76
GPT-4.1	Form 10-Q	4.68	23,970	$0.05162	13.41
Claude Sonnet 4.6	Form 10-Q	4.80	33,664	$0.13339	22.27

3.7 Cost and Latency

Figure 6 presents the API cost distribution across conditions, Figure 7 presents the distribution of wall-clock retrieval time per company, and Figure 8 decomposes mean successful crawl time into fetch time and LLM processing time.

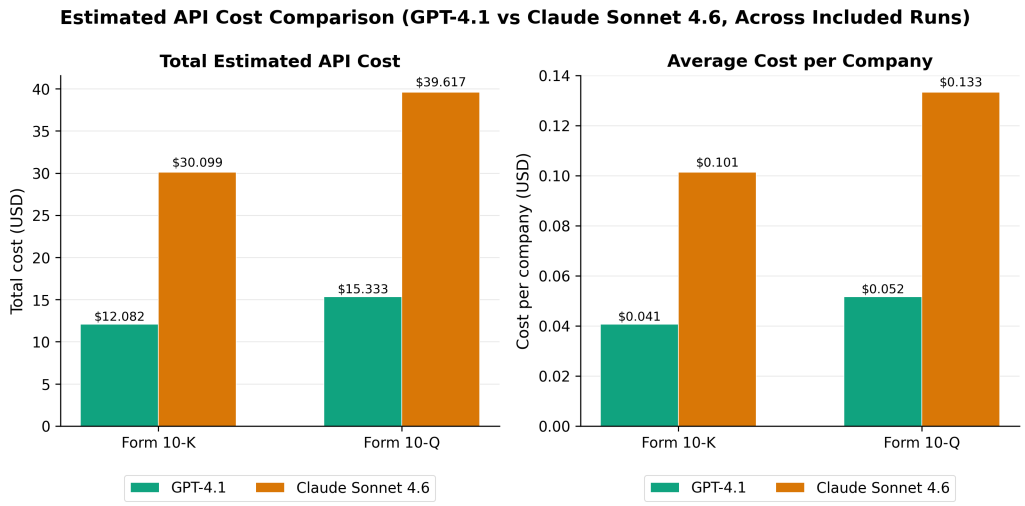


Figure 6: Estimated API cost per condition across the included three-run study.

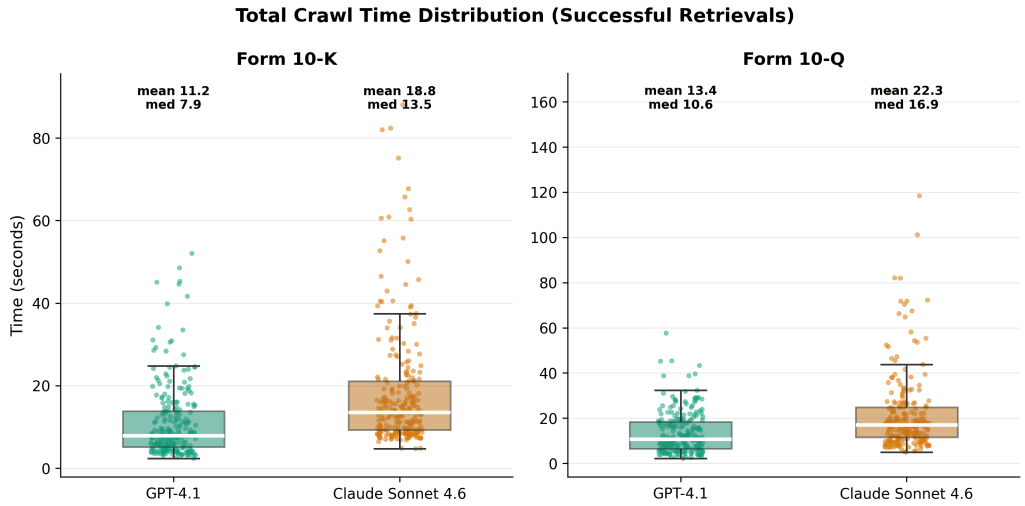


Figure 7: Wall-clock retrieval time per company (seconds) by condition across three runs.

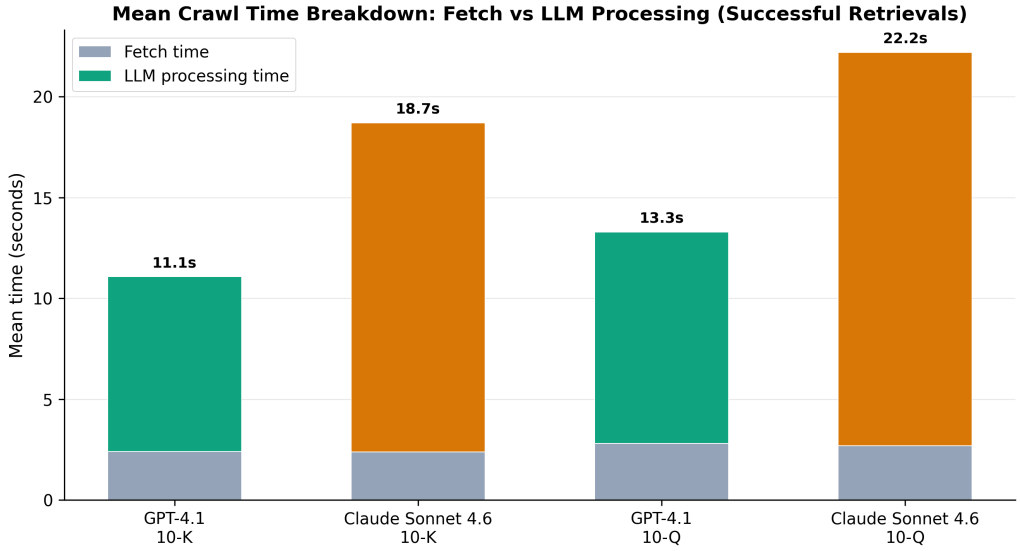


Figure 8: Mean successful crawl time decomposed into fetch time and LLM processing time across three runs.

Claude Sonnet 4.6 is consistently more expensive than GPT-4.1 across both filing types, with the difference driven by higher input-token consumption and longer crawl paths. Mean per-company API cost was \$0.04068 for GPT-4.1 on the 10-K condition compared to \$0.10134 for Claude Sonnet 4.6, a factor of approximately 2.5. On the 10-Q condition the equivalent figures were \$0.05162 and \$0.13339, a factor of approximately 2.6. Claude Sonnet 4.6 was also slower on average: mean wall-clock retrieval time was 11.17 seconds per successful company for GPT-4.1 on 10-K versus 18.76 seconds for Claude, and 13.41 seconds versus 22.27 seconds on the 10-Q condition. The practical implications of these trade-offs are discussed in Section 4.6. Figure 8 shows that this latency gap is driven primarily by LLM processing time rather than by raw fetch time. Mean fetch time is similar across models (approximately 2.4 to 2.8 seconds), whereas mean LLM processing time rises from 8.69 to 16.31 seconds on Form 10-K and from 10.48 to 19.51 seconds on Form 10-Q when moving from GPT-4.1 to Claude Sonnet 4.6.

3.8 Head-to-Head Divergence

Figure 9 presents a head-to-head comparison for the same companies across both models within each filing type, identifying the cases where exactly one model retrieved the correct document and the other did not.

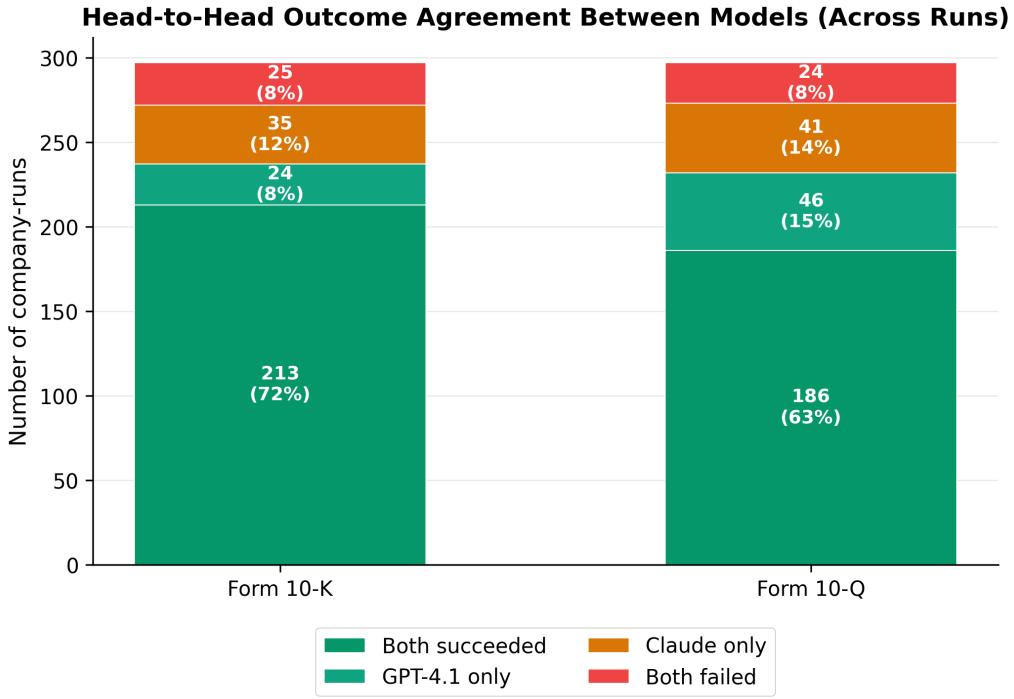


Figure 9: Head-to-head outcome comparison between GPT-4.1 and Claude Sonnet 4.6 per filing type across three runs ($n = 297$ company-runs per filing type).

Across the three 10-K runs, Claude Sonnet 4.6 succeeded where GPT-4.1 did not in 35 observations, while GPT-4.1 succeeded where Claude did not in 24. Both models succeeded on the same company-run in 213 of 297 annual observations (71.7%), and both failed in 25 observations (8.4%). On the 10-Q condition the split narrows and reverses slightly: GPT-4.1 succeeded where Claude did not in 46 observations, compared to 41 observations in the other direction. Both models succeeded in 186 of 297 quarterly observations (62.6%), with 24 shared failures (8.1%). These head-to-head counts suggest that the annual advantage of Claude is material, whereas the quarterly gap is best interpreted as narrow and conditional rather than decisive.

4 Discussion

This section interprets the quantitative results presented in Section 3, examining the performance differences between GPT-4.1 and Claude Sonnet 4.6 across both filing types, the role of autonomous EDGAR navigation as an emergent model behaviour, the impact of site architecture constraints, and the limitations of the study design.

4.1 Form 10-K Retrieval: Claude Sonnet 4.6 is Clearly Stronger

Claude Sonnet 4.6 achieved a mean exact match rate of 80.14% on Form 10-K retrieval across three runs compared to 70.04% for GPT-4.1, a difference of 10.10 percentage points. The head-to-head analysis across 297 annual company-runs shows that this is not a marginal effect: Claude retrieved the correct document in 35 cases where GPT-4.1 did not, while GPT-4.1 retrieved the correct document in only 24 cases where Claude did not.

A major reason for this gap is precision after apparent success. GPT-4.1 returned a non-empty URL in 79.8% of annual attempts, but only 87.8% of those successful retrievals were exact matches. Claude Sonnet 4.6 returned a non-empty URL in 83.5% of annual attempts and converted 96.0% of those successes into exact matches. Inspection of the recurring GPT annual false successes shows a stable pattern of filing-index URLs, submission packages, and alternate hosted artefacts being treated as the final answer. This is a failure of final-document selection and instruction following rather than a failure to reach the relevant filing area.

The three-run data also show that GPT-4.1's 10-K condition is the least stable in the study. Its exact match rate declines from 74.75% in Run 1 to 66.67% in Run 3, producing a standard deviation of 4.20 percentage points. Claude's annual condition also drifts downward, but more moderately (82.83% to 77.78%; SD = 2.54). The important point is that the model ordering does not change: Claude remains ahead on annual retrieval in every run.

This makes the annual conclusion comparatively strong. GPT-4.1 is cheaper, but its annual performance is weakened by both persistent reach failures on JavaScript-heavy portals and a repeatable tendency to stop one step short of the direct filing document.

4.2 Form 10-Q Retrieval: GPT-4.1 is Marginally Stronger, Claude More Stable

On Form 10-Q retrieval the ordering reverses, but only narrowly. GPT-4.1 achieved a mean exact match rate of 74.08% compared to 72.39% for Claude Sonnet 4.6, a difference of 1.69 percentage points. The head-to-head analysis supports the interpretation that this

is a modest edge rather than a decisive one: GPT-4.1 succeeded where Claude did not in 46 quarterly company-runs, while Claude succeeded where GPT-4.1 did not in 41.

Unlike the annual condition, quarterly precision after success is almost identical between the models. GPT-4.1 achieved an exact-match-of-success rate of 94.8%, and Claude 94.7%. The quarterly gap therefore reflects reach, recovery strategy, and cost more than it reflects a large difference in final document selection quality.

Claude Sonnet 4.6 is nevertheless the most stable quarterly condition in the benchmark. Its three exact match rates are effectively flat (72.73%, 71.72%, 72.73%), yielding a standard deviation of only 0.58 percentage points. GPT-4.1's quarterly condition varies more (SD = 2.10), but remains slightly ahead on mean exact match and substantially ahead on cost. The practical quarterly decision is therefore more contingent than the annual one: Claude is behaviourally interesting and stable, but GPT-4.1 offers the better accuracy-to-cost position.

4.3 IR Subdomain Fallback: High Precision, Uneven Reliance

The coded IR subdomain fallback performed strongly across all four conditions. Correctness was 94.4% for GPT-4.1 on Form 10-K, 92.9% for Claude Sonnet 4.6 on Form 10-K, 100.0% for GPT-4.1 on Form 10-Q, and 92.0% for Claude on Form 10-Q. This is important because it shows the fallback itself is not a major source of benchmark noise; when it activates, it usually improves the result.

The more revealing difference is activation frequency. GPT-4.1 required the subdomain fallback 36 times on both annual and quarterly retrieval, whereas Claude required it 28 times on 10-K and 25 times on 10-Q. This suggests that Claude more often reaches the relevant IR area through direct homepage navigation, while GPT relies more heavily on a coded recovery path to recover from shallow homepage navigation failures.

This asymmetry matters for interpreting model differences. The subdomain fallback is a system-level capability available equally to both models, but it compensates more often for GPT-4.1's weaker direct navigation to investor relations content.

4.4 Autonomous LLM Navigation to SEC EDGAR

Neither the system prompt nor any coded logic instructs either model to navigate to SEC EDGAR. The presence of `sec.gov` URLs in the results therefore reflects an autonomous decision by the model to seek the filing from the public EDGAR database when the company IR path is difficult or blocked. This is an unexpected emergent finding of the benchmark rather than a planned system feature.

The behaviour is highly asymmetric between models. On Form 10-K, Claude Sonnet

4.6 navigated to EDGAR 31 times across the three runs and achieved 28 exact matches. GPT-4.1 navigated to EDGAR only 5 times and achieved 0 exact matches. This is one of the clearest behavioural explanations for Claude's annual advantage: it possesses a reliable alternative-source recovery path that GPT largely does not exploit successfully.

On Form 10-Q, Claude also used EDGAR heavily, with 38 EDGAR retrievals and 35 exact matches, while GPT used EDGAR only 6 times with 5 exact matches. After the latest manual validation of alternative SEC URLs in the ground-truth set, Claude's quarterly EDGAR behaviour is better characterised as mostly accurate than it first appeared in earlier drafts. Even so, that strength does not translate into an overall quarterly lead, because the condition remains constrained by persistent reach failures on JavaScript-heavy portals and by Claude's substantially higher cost.

The cross-condition interpretation is therefore nuanced. Claude's EDGAR behaviour is a major annual asset and a valid quarterly recovery mechanism, but its practical value differs by filing type. On 10-K it produces a clear net advantage. On 10-Q it creates an interesting and mostly accurate fallback pathway, yet not enough additional exact matches to outweigh cost and broader site-access constraints.

4.5 JavaScript Rendering as the Dominant Failure Constraint

Across all four conditions, the most persistent site-level cause of failure is JavaScript rendering. The retrieval system uses static HTTP fetching only: pages are retrieved as raw response bodies and converted to Markdown before being passed to the LLM. When filing tables are populated client-side, the static response contains little or no usable filing information, leaving the LLM unable to identify a link regardless of its reasoning quality.

This pattern is visible in the recurring failure sets across runs. Major companies such as Alphabet, Amazon, Apple, NVIDIA, and AT&T repeatedly fail for one or both models, and the associated benchmark traces are dominated by `none_action`, `fetch_failed`, or other access-related terminations rather than by plausible-but-wrong final documents. The appendix examples in Appendix C are important because they show that some IR portals are effectively invisible to a static crawler even when the page appears usable in a real browser.

This finding places a ceiling on the benchmark. The observed differences between GPT-4.1 and Claude Sonnet 4.6 occur within the subset of sites that are statically accessible enough for either model to act on. Model choice can shift performance within that accessible subset, but it cannot fully overcome JavaScript-dependent site architecture.

4.6 Cost and Latency Trade-offs

GPT-4.1 is materially more cost-efficient than Claude Sonnet 4.6 across both filing types. Mean per-company API cost was $0.04068 for GPT-4.1 on the 10-K condition compared to $0.10134 for Claude, a factor of approximately 2.5. On the 10-Q condition the equivalent figures were $0.05162 and $0.13339, a factor of approximately 2.6.

GPT-4.1 was also faster on average. Mean successful crawl time was 11.17 seconds for GPT-4.1 versus 18.76 seconds for Claude on Form 10-K, and 13.41 seconds versus 22.27 seconds on Form 10-Q. The difference is consistent with Claude's greater token consumption and its tendency to pursue deeper or more exploratory navigation paths, including frequent EDGAR fallback.

These trade-offs matter because they prevent a one-dimensional recommendation. For organisations where annual filing accuracy is the dominant requirement, Claude Sonnet 4.6 offers a meaningful performance gain. For cost-sensitive deployments, or for workflows dominated by quarterly retrieval, GPT-4.1 offers a more attractive balance of price, latency, and accuracy.

4.7 Limitations

Several limitations of this study should be acknowledged. First, the benchmark is based on three completed runs per condition rather than four or more. Three runs are sufficient to estimate a mean and standard deviation and to identify stable versus unstable conditions, but a larger number of repeated runs would provide tighter confidence around the observed differences, especially for the more variable annual conditions.

Second, the static-fetch design described in Section 2.3 means that the results cannot be generalised directly to systems that incorporate browser automation. The benchmark measures performance within a specific architectural constraint, and the results should be interpreted accordingly.

Third, ground truth URLs were manually verified at a specific point in time. Corporate IR sites change over time, and the latest validation pass in this study confirmed that a small number of quarterly SEC URLs should be treated as valid alternatives. This underlines that accurate benchmarking depends not only on model output but also on continued maintenance of the ground-truth dataset.

Fourth, URL normalisation was limited to scheme standardisation and trailing slash removal. More complex URL equivalences, such as redirect chains or query parameter reordering, were not treated as exact matches, which may have caused a small number of legitimate near-equivalences to be scored as mismatches.

4.8 Threats to Validity

With respect to internal validity, the task prompts were developed iteratively during system construction and may reflect assumptions about filing presentation that align more naturally with one model's behaviour than the other's. Although the same prompts were applied to both models, the possibility of inadvertent prompt bias cannot be entirely excluded.

With respect to external validity, the dataset comprises 99 US-listed companies subject to SEC filing requirements. The results may not generalise to non-US regulatory filing contexts, to smaller companies with less professionally maintained IR sites, or to filing types other than the 10-K and 10-Q. The benchmark reflects a snapshot of IR site conditions during the period in which the runs were conducted, and performance may vary across different time periods as sites are updated.

5 Conclusion

5.1 Summary of Findings

This study conducted a controlled quantitative benchmark comparing GPT-4.1 and Claude Sonnet 4.6 on the task of automated retrieval of SEC financial filings from corporate investor relations websites. Four conditions were evaluated across three independent runs each, covering two models and two filing types (Form 10-K and Form 10-Q), with 99 US-listed companies per run and 1,188 company-runs overall.

Three principal findings emerge. First, Claude Sonnet 4.6 is the stronger model for Form 10-K retrieval, achieving a mean exact match rate of 80.14% compared to 70.04% for GPT-4.1, a gap of 10.10 percentage points. Claude's advantage is driven by a lower false-success burden and by its autonomous navigation to SEC EDGAR, which acted as a reliable alternative-source recovery path on annual filings. Second, GPT-4.1 is the slightly stronger model for Form 10-Q retrieval, achieving a mean exact match rate of 74.08% compared to 72.39% for Claude Sonnet 4.6. The quarterly gap is modest, but GPT-4.1 pairs that small accuracy advantage with materially lower cost. Third, JavaScript rendering is the dominant site-level failure constraint for both models. Companies whose IR portals rely on client-side JavaScript to populate filing listings remain inaccessible to a static-fetch retrieval system regardless of which model is used, imposing a performance ceiling that model choice alone cannot remove.

A further behavioural distinction is that Claude Sonnet 4.6 showed stronger emergent alternative-source behaviour, most notably its autonomous use of SEC EDGAR when direct IR-site navigation failed. On the 10-K condition this behaviour materially improved retrieval performance. On the 10-Q condition it was also mostly accurate after final ground-truth validation, but still did not create a practical quarterly lead because reach failures and cost remained the binding constraints.

Taken together, the four research questions can be answered directly. Claude Sonnet 4.6 is superior for Form 10-K retrieval, GPT-4.1 is marginally superior for Form 10-Q retrieval, JavaScript-rendered IR architecture is the strongest site-level predictor of failure across both models, and the final deployment recommendation depends on whether annual accuracy or operational cost and latency are the dominant priority.

5.2 Recommendations

The following recommendations are made for the production system design decision at the firm.

For Form 10-K retrieval where accuracy is the primary concern: Claude Sonnet

4.6 is recommended. Its mean exact match rate of 80.14% is 10.10 percentage points higher than GPT-4.1, and its exact-match-of-success rate of 96.0% means it is much less likely to return an incorrect annual document after apparently succeeding. Its stronger emergent use of EDGAR as an alternative authoritative source is also a meaningful advantage when conventional IR navigation fails. The higher API cost is justified where annual retrieval accuracy directly affects the quality of downstream research.

For Form 10-Q retrieval, or for cost-sensitive deployments across both filing types: GPT-4.1 is the more pragmatic choice. It achieves a slightly higher mean exact match rate on quarterly filings (74.08% vs 72.39%) while costing roughly 2.5 times less per company-run than Claude on average across the study. The quarterly accuracy advantage should be interpreted as modest rather than decisive, but the combined cost, latency, and accuracy profile still favours GPT-4.1 for routine large-scale use.

For deployments requiring high recall on JavaScript-heavy IR sites: Neither model in its current configuration is sufficient. Browser automation via a tool such as Playwright should be investigated as a supplementary fetching strategy for the subset of companies whose IR portals are identified as JavaScript-rendered. A hybrid architecture, using static fetching as the primary path and browser automation as a targeted fallback, would address the dominant failure class identified in this study without incurring the full latency overhead of browser-based fetching for all companies.

5.3 Future Work

Several directions for further investigation follow from this study.

A fourth benchmark run for each condition would strengthen confidence in the observed means, particularly for the annual conditions where variability is higher than on the quarterly benchmark. A fourth run is not necessary to establish the main model ordering reported here, but it would help quantify whether the downward drift observed in both 10-K conditions persists.

Browser-based fetching as a fallback for JavaScript-rendered IR pages warrants investigation. Playwright or an equivalent tool could be invoked selectively for companies where the static fetcher returns an empty or near-empty Markdown document, rather than being applied universally. Benchmarking this hybrid approach against the current static-only system would quantify the accuracy gain relative to the latency cost.

The benchmark dataset could be extended beyond 99 companies, with deliberate inclusion of companies known for complex or JavaScript-heavy IR architectures, to produce a more demanding evaluation of model robustness. Extension to non-US companies would also test whether the findings generalise to the global deployment context for which the system

is ultimately intended.

Finally, per-model prompt optimisation and dynamic routing have not been explored in this study. A production system could, for example, use the cheaper GPT-4.1 path by default while escalating difficult annual cases to Claude Sonnet 4.6 or to a browser-assisted fallback when early signals indicate a static-accessibility problem.

Declaration of Academic Integrity

I, Daniel Moody, declare that this report is entirely my own work unless otherwise stated and referenced. I have read and understood the University of Limerick policy on academic integrity and plagiarism. All sources used in the preparation of this report have been acknowledged through proper citation in accordance with the UL Cite It Right referencing standard. No part of this report has been submitted for any other assessment at this or any other institution.

AI Use Declaration

The use of generative artificial intelligence is an integral and explicit component of this project. The two AI systems evaluated in this study, OpenAI GPT-4.1 and Anthropic Claude Sonnet 4.6, are the subjects of the benchmarking investigation. Both were used solely as the tools under test: they were invoked programmatically via their respective APIs to perform automated web navigation and document retrieval tasks, and their outputs were evaluated against a manually verified ground truth dataset. Neither model was used to write, draft, or edit any part of this report.

This declaration is made in accordance with the ISE FYP 2025–2026 specification, which permits the use of generative AI where it is an integral part of the project and is discussed in full within the report.

References

Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., Rosenberg, M., Song, X., Stoica, A., Tiwary, S. & Wang, T. (2016), 'MS MARCO: a human generated machine reading comprehension dataset', *arXiv preprint arXiv:1611.09268* . available: doi: 10.48550/arXiv.1611.09268.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G. et al. (2021), 'Evaluating large language models trained on code', *arXiv preprint arXiv:2107.03374* . available: doi: 10.48550/arXiv.2107.03374.

Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H. & Su, Y. (2023), Mind2Web: towards a generalist agent for the web, *in* 'Advances in Neural Information Processing Systems'. available: doi: 10.48550/arXiv.2306.06070.

Frolov, S. & Wustrow, E. (2019), The use of TLS in censorship circumvention, *in* 'Proceedings of the Network and Distributed Systems Security Symposium'. available: `https://www.ndss-symposium.org/ndss-paper/the-use-of-tls-in-censorship-circumvention/` [accessed 22 Mar 2026].

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D. & Steinhardt, J. (2021), Measuring massive multitask language understanding, *in* 'International Conference on Learning Representations'. available: doi: 10.48550/arXiv.2009.03300.

Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A. et al. (2022), Holistic evaluation of language models, *in* 'Advances in Neural Information Processing Systems'. available: doi: 10.48550/arXiv.2211.09110.

Loughran, T. & McDonald, B. (2011), 'When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks', *The Journal of Finance* **66**(1), 35–65. available: doi: 10.1111/j.1540-6261.2010.01625.x.

Mesbah, A., van Deursen, A. & Lenselink, S. (2012), 'Crawling Ajax-based web applications through dynamic analysis of user interface state changes', *ACM Transactions on the Web* **6**(1), 1–30. available: doi: 10.1145/2109205.2109208.

Mirtaheri, S. M., Dinçtürk, M. E., Hooshmand, S., von Bochmann, G., Jourdan, G.-V. & Onut, I.-V. (2013), A brief history of web crawlers, *in* 'Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research', pp. 40–54. available: doi: 10.48550/arXiv.1405.0749.

Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W. et al. (2021), 'WebGPT: browser-assisted question-

answering with human feedback', *arXiv preprint arXiv:2112.09332* . available: doi: 10.48550/arXiv.2112.09332.

U.S. Securities and Exchange Commission (2024), 'EDGAR: Electronic Data Gathering, Analysis, and Retrieval system', available: `https://www.sec.gov/search-filings`. [accessed 24 Mar 2026].

Wu, S., Irsoy, O., Lu, S., Dabravolski, V., Dredze, M., Gehrmann, S., Kambadur, P., Rosenberg, D. & Mann, G. (2023), 'BloombergGPT: a large language model for finance', *arXiv preprint arXiv:2303.17564* . available: doi: 10.48550/arXiv.2303.17564.

Yang, Y., Uy, M. C. S. & Huang, A. (2020), 'FinBERT: a pretrained language model for financial communications', *arXiv preprint arXiv:2006.08097* . available: doi: 10.48550/arXiv.2006.08097.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. & Cao, Y. (2023), ReAct: synergizing reasoning and acting in language models, *in* 'International Conference on Learning Representations'. available: doi: 10.48550/arXiv.2210.03629.

A Prompt Templates

The two document query definitions below specify the exact instructions provided to both models for each filing type. The `DocumentQuery` dataclass contains the document type label, a recency filter, a navigation path, and free-text disambiguation hints. These definitions were applied identically to GPT-4.1 and Claude Sonnet 4.6 across all runs.

Listing 1: DocumentQuery and NavigationTarget dataclass definitions.

```
@dataclass(frozen=True)
class NavigationTarget:
    description: str
    hints: str = ""

@dataclass(frozen=True)
class DocumentQuery:
    start_url: str
    nav_path: list[NavigationTarget]
    document_type: str
    document_filter: str
    hints: str = ""
```

Listing 2: Form 10-K query definition with disambiguation hints.

```
FORM_10K = DocumentQuery(
    start_url="",
    nav_path=[NavigationTarget("investor relations page")],
    document_type="Form 10-K",
    document_filter="most recent",
    hints=(
        "The target is the SEC Form 10-K (the regulatory annual "
        "filing), NOT the glossy shareholder Annual Report PDF. "
        "Look for links labeled '10-K', 'Form 10-K', "
        "'Annual Report on Form 10-K', or inside an "
        "'SEC Filings' section. "
        "Do NOT select a link labeled only 'Annual Report' "
        "unless it explicitly says 'Form 10-K'. Usually the "
        "annual report is on a main page or within a "
        "SEC FILINGS section"
    ),
)
```

Listing 3: Form 10-Q query definition with disambiguation hints.

```
FORM_10Q_RECENT = DocumentQuery(
    start_url="",
    nav_path=[NavigationTarget("investor relations page")],
    document_type="Form 10-Q",
    document_filter="most recent",
    hints=(
        "The target is the SEC Form 10-Q (the regulatory "
        "quarterly filing). Look for links labeled '10-Q', "
        "'Form 10-Q', usually these can be found by navigating "
        "to a 'SEC Filings' section or a reports/corporate "
        "filings section. "
        "Do NOT select earnings press releases, investor "
        "presentations, or earnings call transcripts. "
```

```
        "Do NOT select a Form 10-K -- that is the annual "
        "filing, not a quarterly 10-Q."
    ),
)
```

B Benchmark Runner

The benchmark runner processes companies sequentially, with each company crawled independently using a fresh HTTP fetcher and instrumentation context. Results are written to disk after each company to prevent data loss. Up to three automatic retry attempts are made on exception, separated by a three-second delay.

Listing 4: Benchmark runner: CLI interface and condition setup.

```
MODELS = {
    "gpt": lambda: OpenAIClient(),
    "claude": lambda: ClaudeClient(),
}

DOC_TYPES = {
    "10k": FORM_10K,
    "10q": FORM_10Q_RECENT,
}
```

Listing 5: Per-company crawl function with retry logic and URL normalisation.

```
async def crawl_one(bc, llm_client, query_template,
                    doc_key, index, total):
    company = bc.company
    expected_url = (bc.expected_10k_url if doc_key == "10k"
                    else bc.expected_10q_url)
    alt_url = (bc.alt_10k_url if doc_key == "10k"
               else bc.alt_10q_url)

    max_retries = 3
    for attempt in range(max_retries):
        metrics = BenchmarkMetrics()
        timed_fetcher = TimedFetcher(
            CffiHttpFetcher(), metrics.timings)
        timed_llm = TimedLLMClient(
            llm_client, metrics.timings)

        t0 = time.perf_counter()
        try:
            async with CompanyDocumentCrawler(
                timed_llm, fetcher=timed_fetcher
            ) as crawler:
                result, discovery = (
                    await crawler.find_with_details(
                        company, query_template))
            metrics.total_elapsed_s = (
                time.perf_counter() - t0)
            status = ("success" if result.document_url
                      else "failed")
            break
```

```
        except Exception as e:
            if attempt < max_retries - 1:
                await asyncio.sleep(3)

    # URL normalisation and ground truth comparison
    def _normalise(url):
        return url.replace(
            "http://", "https://").rstrip("/")

    norm_doc = _normalise(doc_url) if doc_url else ""
    url_match = bool(norm_doc and (
        (expected_url
         and norm_doc == _normalise(expected_url))
        or (alt_url
            and norm_doc == _normalise(alt_url))
    ))
```

Listing 6: Main benchmark loop: sequential processing with incremental save.

```
async def run_benchmark(model_key, doc_key,
                        run_number, limit):
    llm_client = MODELS[model_key]()
    query_template = DOC_TYPES[doc_key]
    companies = load_benchmark_companies(
        BENCHMARK_CSV, limit=limit)

    results = []
    prev_input, prev_output = 0, 0

    for i, bc in enumerate(companies, 1):
        row = await crawl_one(
            bc, llm_client, query_template,
            doc_key, i, len(companies))

        # Per-company token delta
        curr_input = llm_client.token_usage.input_tokens
        curr_output = llm_client.token_usage.output_tokens
        row["token_usage"] = {
            "input_tokens": curr_input - prev_input,
            "output_tokens": curr_output - prev_output,
            "total_tokens": ((curr_input - prev_input)
                + (curr_output - prev_output)),
        }
        prev_input, prev_output = curr_input, curr_output

        results.append(row)
        # Write after each company to prevent data loss
        results_file.write_text(
            json.dumps(results, indent=2))
```

C JavaScript Failure Evidence

The figures below illustrate the two categories of JavaScript rendering failure observed in the benchmark. In both cases the static HTTP fetcher returns a response body that contains no navigable filing links, making correct retrieval impossible regardless of model

capability.

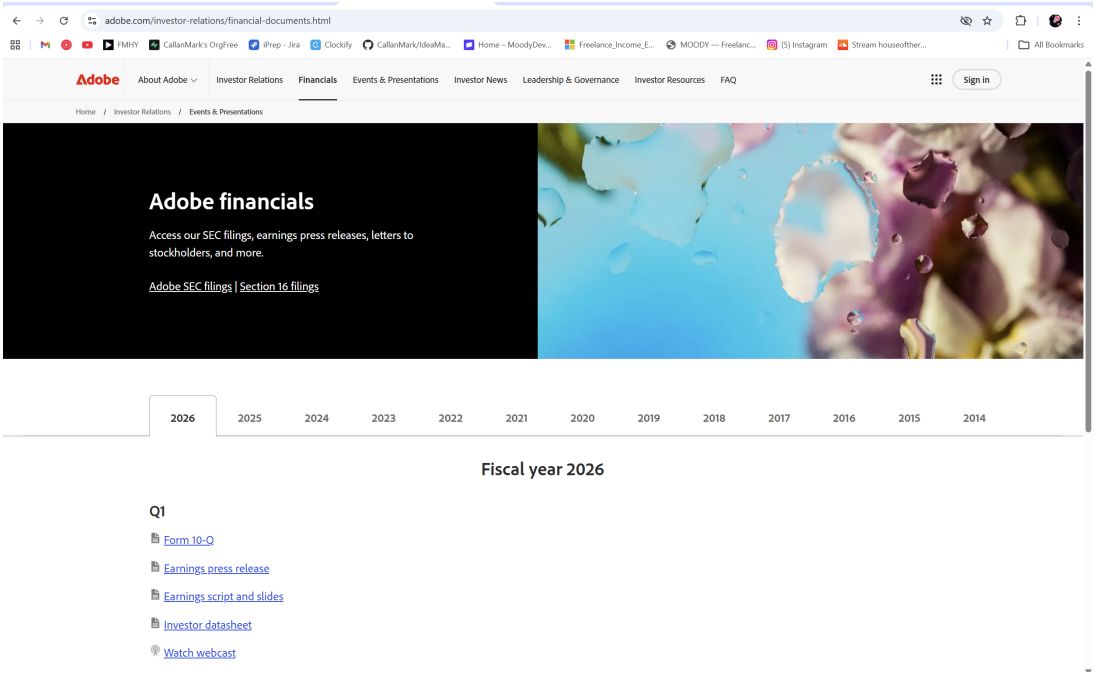


Figure 10: Adobe Inc. IR portal: the filing listing is rendered entirely by a JavaScript single-page application. The static HTTP response body contains no document links; the LLM receives an effectively empty page and terminates with a `none_action` outcome.

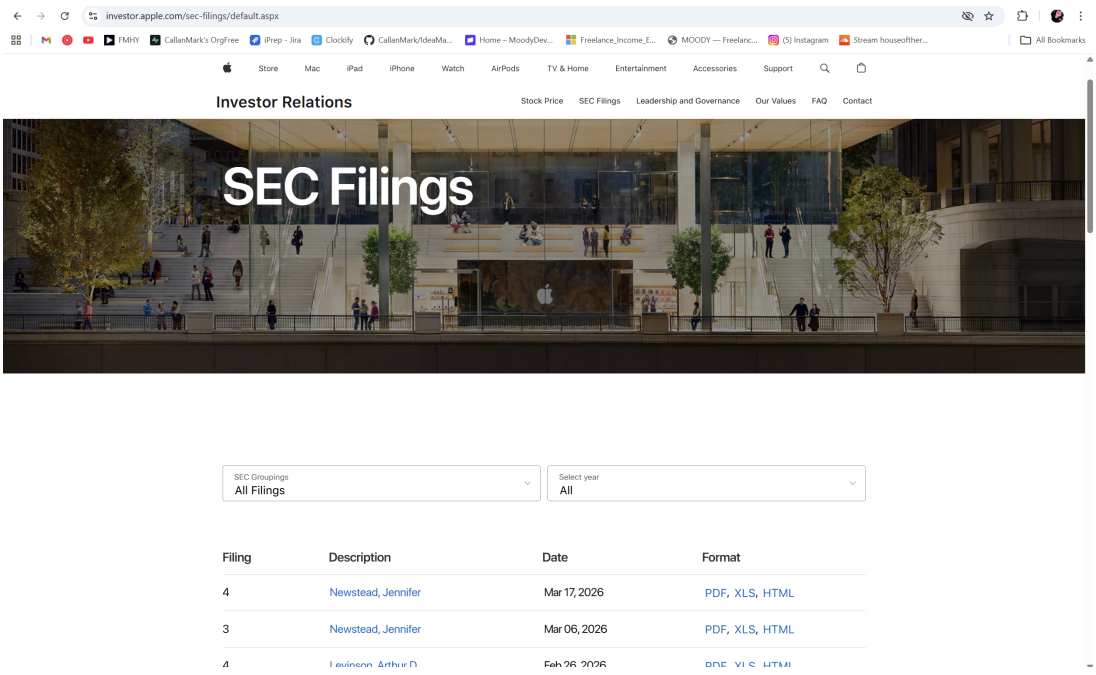


Figure 11: Apple Inc. IR portal: the filing table is populated via an asynchronous JavaScript API call after page load. The static response body contains the table container element but no filing rows, leaving the LLM with no document links to follow.